EL961413932

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Custom Emoticons

Inventor:
Bettina Walter
Jens Martin von Seelen Thorsen

ATTORNEY'S DOCKET NO. MS1-1801US

TECHNICAL FIELD

The subject matter relates generally to instant messaging and more specifically to custom emoticons.

BACKGROUND

Emoticons have acquired immense popularity and hence importance in new email, chatroom, instant messaging, and even operating system applications. The variety of available emoticons has increased tremendously, from a few types of "happy faces" to a multitude of elaborate and colorful animations. In many cases, an increase in the number of available emoticons has been a selling feature for new releases of communications products. However, there are now so many emoticons available that some applications may be reaching a limit on the number of pre-established ("pre-packaged") emoticons that can be included with or managed by an application. There is an exhaustion point for trying to provide a pre-packaged emoticon for every human emotion. Still, users clamor for more emoticons, and especially for more nuanced emoticons that capture the subtleties of human emotions and situations.

Fig. 1 shows a conventional list of pre-packaged emoticons 100 in a dialogue box of an application. Typically a user selects one of the emoticons to insert within a textual dialogue or email by double clicking a mouse on an icon 102 of the emoticon, or by clicking on the icon 102 and then actuating a select button 104.

With the increase in the number and "sophistication" of emoticons, some problems are inevitable. Many chatroom and instant messaging products, in which emoticons find perfect application in describing the users' real-time

emotions, rely on limited bandwidth or the transmission of short communications ("messages") that contain relatively little data. In other words, many chatroom and instant messaging applications achieve their agility and speed by streamlining data bulk into "lean" messages that typically have a limited data size, such as 1-2 kilobytes or approximately 400 alphanumeric characters plus headers. Adding one or more complex emoticons—a graphic that may require an inordinate amount of data space compared with text—to one of these lean messages can be detrimental to the performance of the chatroom or instant messaging application. Still, an emoticon picture is often worth a thousand words of text, so techniques are needed for producing an even greater variety of emoticons and for being able to send them without increasing the data size of lean messages.

SUMMARY

Methods and devices for creating and transferring custom emoticons allow a user to adopt an arbitrary image as an emoticon, which can then be represented by a character sequence in real-time communication. In one implementation, custom emoticons can be included in a message and transmitted to a receiver in the message. In another implementation, character sequences representing the custom emoticons can be transmitted in the message instead of the custom emoticons in order to preserve performance of text messaging. At the receiving end, the character sequences are replaced by their corresponding custom emoticons, which can be retrieved locally if they have been previously received, or can be retrieved from the sender in a separate communication from the text message if they have not been previously received.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a graphic representation of conventional pre-packaged emoticons available in an application.

Fig. 2 is a graphic representation of exemplary custom emoticons displayed in a text message but not transferred with the text message.

Fig. 3 is a block diagram of an exemplary custom emoticon engine.

Fig. 4 is a block diagram of a message receiver module of an exemplary custom emoticon engine.

Fig. 5 is a graphic representation of an exemplary list of custom emoticons.

Fig. 6 is a graphic representation of an exemplary dialogue box for defining custom emoticons.

Fig. 7 is a flow diagram of an exemplary method of creating and sending custom emoticons.

Fig. 8 is a flow diagram of an exemplary method of receiving a real-time communication that includes custom emoticons.

Fig. 9 is a block diagram of an exemplary computing device environment for practicing the subject matter.

DETAILED DESCRIPTION

Overview

The subject matter described herein includes methods and devices for creating custom emoticons from images not previously associated with emoticons. Further, when custom emoticons are added to a text message, some implementations of the subject matter transmit the custom emoticons to a receiver of the message without degrading the textual performance of instant messaging applications, chatroom exchanges, and toasts—the notifications that appear when buddies log on.

Keyboard keystrokes or textual alphanumeric "character sequences" are assigned as placeholders for custom emoticons within a real-time message. A custom emoticon or its associated placeholder character sequence can be entered in an appropriate location of a real-time message during composition of the message. Custom emoticons present in the message are typically converted to their corresponding character sequences before transmitting the message in order to preserve the performance of text messaging. The character sequences map to their associated custom emoticons. Upon receiving a real-time message that includes an assigned character sequence, the character sequence can be mapped back to a corresponding custom emoticon, which can then be displayed in the received message in place of the character sequence. Thus, custom emoticons do not necessarily have to be sent within the text message itself. This allows text messaging to remain lean and fast, while the subject matter provides ways for reconstructing custom emoticons into the message at the receiving end.

Instead of selecting from a necessarily limited host of pre-packaged emoticons, users can create their own emoticons by adapting many sorts of image

files to be custom emoticons. In one implementation, image files of various types and sizes are each standardized into a pixel array of uniform dimensions to be used as emoticons.

Many real-time messaging applications aim to minimize data for transmission. An instant message or chatroom communiqué that contains very streamlined data is referred to herein as a "lean" message. To include one or more custom emoticons in a lean message would degrade the performance of many instant messaging or chatroom applications. Thus, the subject matter also includes exemplary techniques for sending a lean message to a receiver wherein one or more custom emoticons appear in the lean message at the receiver's end.

"Real-time" as used herein means that participants can converse back and forth via text, image, or sound without unreasonable delay while they are online. Thus, real-time technically means "near real-time" as there is always some degree of hysteresis or delay in online communication.

Reference in this specification to "one implementation" or "an implementation" means that a particular feature, structure, or characteristic described in connection with the implementation is included in at least one implementation of the subject matter. The appearances of the phrase "in one implementation" in various places in the specification are not necessarily all referring to the same implementation.

Exemplary System

Fig. 2 shows an exemplary system 200 for creating and transferring custom emoticons. Multiple network nodes (e.g., 202, 204) are communicatively coupled so that users may communicate using instant messaging, a chatroom application,

email, etc. In one implementation, each node 202, 204 includes an exemplary custom emoticon engine "CEE" (e.g., 206, 208). An exemplary CEE 206 allows a user to adopt an arbitrary image 210 as a custom emoticon. An exemplary CEE 206 also performs real-time mapping from the text of a typed message to custom emoticons to be included, inserted, substituted, etc., into the message, without including the custom emoticons in the transmission of the text message itself.

A user can create a custom emoticon by downloading and/or editing an image, or by using a photography or drawing application to create an image for the custom emoticon from scratch. Once a user has adopted an arbitrary image 210 to be a custom emoticon, an exemplary CEE 206 allows the user to send a text message 212 that contains only text and yet have one or more custom emoticons 210 appear at appropriate places in the display of the text message 212'.

Exemplary Engine

Fig. 3 shows an exemplary custom emoticon engine, CEE 206, in greater detail than in Fig. 2. An exemplary CEE 206 typically resides on clients 202, 204, that is, on a message sender's computing device 202 and a message receiver's computing device 204. Of course the sending and receiving roles change frequently during real-time communication between two or more users. Thus, an exemplary CEE 206 has components for sending text messages and custom emoticons and components for receiving text messages and custom emoticons. An exemplary computing device environment suitable for an exemplary CEE 206 and suitable for practicing exemplary methods described herein is described with respect to Fig. 9.

An exemplary CEE 206 includes a user interface 302 that may include a "define custom emoticons" module 304 and a "create text message" module 306. The exemplary CEE 206 also includes an image selector 308, a custom emoticons object store 310, and a transmitter 312, all communicatively coupled as illustrated. The image selector 308 may also include a pixel array generator 314. Further, an exemplary CEE 206 may include a character sequence assignor 316 and a message receiver 318 that includes a character sequence parser 320. Finally, an exemplary CEE 206 may include a text editor 322, either discretely as illustrated or integrated with the user interface 302, and a message transmitter 324 that may include a header engine 326. Control logic (not shown) may direct operation of the various components listed above and associations between the components.

As controlled by an automatic process or by a user through a "define custom emoticons" dialogue generated by a module (304) of the user interface 302, an image selector 308 captures an image and converts the image to an emoticon. In one implementation, images of various sizes and formats, such as the joint photographic experts group (JPEG) format, the tagged image file format (TIFF) format, the graphics interchange format (GIF) format, the bitmap (BMP) format, the portable network graphics (PNG) format, etc., can be selected and converted into custom emoticons by a pixel array generator 314, which converts each image into a pixel array of pre-determined dimensions, such as 19 x 19 pixels. An image may be normalized in other ways to fit a pre-determined pixel array grid. For example, if the pre-determined pixel array for making a custom emoticon is a 19 x 19 pixel grid, then the aspect ratio of an image that does not fill the grid can be maintained by adding background filler to the sides of the image to make up the 19 x 19 pixel grid.

In one implementation, an exemplary CEE 206 also includes advanced image editing features to change visual characteristics of an adopted image so that the image is more suitable for use as a custom emoticon. For example, an advanced image editor may allow a user to select the lightness and darkness, contrast, sharpness, color, etc. of an image. These utilities may be especially useful when reducing the size of a large image into a pixel array dimensioned for a modestly sized custom emoticon.

Each new custom emoticon can be saved in a custom emoticons object store 310 together with associated information, such as a character sequence for mapping from a text message to the emoticon and optionally, a nickname. In one implementation, a nickname serves as the mapping character sequence, so that a custom emoticon is substituted for the nickname each time the nickname appears in a received text message. In other implementations, a unique character sequence assigned to a custom emoticon maps to the custom emoticon and a nickname is associated with the custom emoticon as well, so that the nickname or common name may be used in a text message without calling for insertion of a custom emoticon.

The character sequence assignor 316 may utilize a "define custom emoticons" dialogue (304) or an automatic process to associate a unique "character sequence" with each custom emoticon. A character sequence usually consists of alphanumeric characters (or other characters or codes that can be represented in a text message) that can be typed or inserted by the same text editor 322 that is creating a text message. Although keystrokes imply a keyboard, other conventional means of creating a text message can also be used to form a character sequence of characters or codes to map to a custom emoticon.

In one implementation, character sequences are limited to a short sequence of characters, such as seven. The character sequence "dog" can result in a custom emoticon of a dog appearing each time "dog" is used in a message, so other characters may be added to common names to set mappable character sequences apart from text that does not map to a custom emoticon. Hence a character sequence may use brackets, such as [dog] or an introductory character, such as @dog.

When a character sequence has been assigned to a custom emoticon by a character sequence assignor 316, the custom emoticon or its assigned character sequence may by used in an online real-time message. If the custom emoticon itself is embedded in a text message during composition of the message, then the custom emoticon is typically converted to its assigned character sequence before transmission to a message receiver. In some implementations, this allows an exemplary CEE 206 to maintain the speed and other performance characteristics of text messaging if custom emoticons occupy enough data space to degrade performance when transmitted along with a corresponding text message. Thus, a message transmitter 324 may be employed to send the character sequence to a destination, i.e., within a communication such as an instant message. Each time the character sequence associated with a custom emoticon is used in a text message, a custom emoticon associated with the character sequence will be substituted for the character sequence at the destination, e.g., the receiving client 204.

When a user creates a text message, for example using a "create text message" dialogue (306) associated with a user interface 302 and/or a text editor 322, the text message may contain one or more of the character sequences that

map to custom emoticons. In one implementation, a message transmitter 324 or other component of an exemplary CEE 206 may pass the text message to a character sequence parser 320 to find character sequences within the text message that map to custom emoticons. The character sequences may be used in various ways, depending on the implementation.

In one implementation, a character sequence assigned to a custom emoticon may be made mappable to the custom emoticon by parsing the character sequence into an object name that includes both a location of the associated custom emoticon and a hash and/or a globally unique identifier (GUID). The location can be used by a message recipient to retrieve the associated custom emoticon from the sender and the hash and/or GUID can be used both on sending and receiving ends for local storage. Retrieval of custom emoticons pointed to by associated character sequences in a real-time message will be discussed in greater detail in relation to Fig. 4. It should be noted that "retrieving" a custom emoticon, pixels representing a custom emoticon, or data representing the pixels includes a sender, a remote store, or a local store "transmitting" the custom emoticon, the pixels, or the data to the client or process performing the retrieving.

In another implementation, the custom character sequences themselves may be placed in a header of an outgoing text message by a header engine 326, to be read by a receiving client 204 in order to request custom emoticons from the sending client 202. In yet another implementation, instead of or in addition to placing the custom emoticon information in a header of the outgoing text message, the character sequence parser 320 may inform the custom emoticons object store 310 to immediately begin sending the custom emoticons corresponding to the exemplary character sequences found in the text message to the recipient by

another communications pathway. In all three of the implementations just described, however, the text message is sent by a message transmitter 324 using one modality of transmission, such as an instant messaging application, and the custom emoticons are sent by a separate modality, such as a transmitter 312 using an object store and/or an object transport mechanism, in order to spare an increase in the data size of the text messages sent by the message transmitter 324.

It should be noted that an exemplary CEE 206 can be implemented in software, firmware, hardware, or a combination of software, firmware, hardware, etc. The illustrated exemplary CEE 206 is only one example of software, firmware, and/or hardware that can perform the subject matter.

Fig. 4 shows the message receiver 318 of Fig. 3 in greater detail. A message receiver 318 may include a message buffer 402, header parser 404, character sequence parser 320, emoticon retriever 406, and substitution module 408, communicatively coupled as illustrated. The emoticon retriever 406 may further include a local cache retriever 410, a point-to-point retriever 412, and a server retriever 414.

In one implementation, when a user receives a communication, such as a chatroom communication or an instant message, the message buffer 402 holds the message for display. In one implementation, the message is displayed immediately whether custom emoticons are available to be posted within the message or not. If they are not ready, then the original character sequences that map to the custom emoticons may be displayed, or a blank image may be displayed into which a custom emoticon is inserted when it has been retrieved. In another implementation, a message is not displayed at all until relevant custom emoticons are ready to be substituted into the message, but since the time interval

for retrieving custom emoticons is usually unnoticeable, this "delay" in "real-time" communication may also be unnoticeable.

In some implementations, a received message may have one or more headers to be parsed by the header parser 404 that indicate the presence of one or more custom emoticons in a message. A header can also indicate a location for retrieving custom emoticons, and in some implementation, can even indicate the position of custom emoticons to be inserted within the received message.

In one implementation, the message receiver 318 uses an object name provided by the header parser 404 for each custom emoticon to be retrieved and then inserted in a received message. That is, a character sequence parser 320 on the sender's computing device may have already parsed a mappable character sequence (representing a custom emoticon data object) into an object name, including a location identifier and a hash value, and placed the object name in a header of the message sent to the receiving client 204. The object name allows retrieval of the custom emoticon from a location specified by the location identifier for storage in a local cache using the hash value as a storage address. An implementation of the message receiver 318 (and an exemplary CEE 206) may use object store and object transport mechanisms and technologies as described in U.S. Patent Application No. 10/611,599 by Miller et al., entitled "Instant Messaging Object Store" and U.S. Patent Application No. 10/611,575 by Miller et al., entitled "Transport System for Instant Messaging," which are incorporated herein by reference for all they disclose and teach.

In other implementations, instead of or in addition to using the object store and transport technologies mentioned above, the message receiver 318 uses a character sequence parser 320 of an exemplary CEE 206 on the receiving client's

computing device to obtain character sequences from the message in order to map to custom emoticons. In some implementations, the mappable character sequences may be collected in a header of the message, but in still other implementations the character sequence parser 320 performs in-string functions or otherwise searches the message for mappable character sequences.

An emoticon retriever 406 receives an emoticon identifier, such as an object name, from a header parser 404 or a mappable character sequence from either the character sequence parser 320 or the header parser 404, and uses the object name or the character sequence to map to and retrieve a custom emoticon. In one implementation, the custom emoticon is transferred as a PNG file. In one implementation, the emoticon retriever 406 first activates a local cache retriever 410, for example using the above-mentioned hash component of an object name. The custom emoticon being retrieved may have previously been retrieved from the message sender and may now reside in a local cache, such as a temporary Internet files cache used by a web browser. If the custom emoticon resides locally, the custom emoticon can be immediately loaded and/or retrieved as an image file. In one implementation a local cache retriever 410 of an exemplary CEE 206 uses a globally unique identifier (GUID) when checking to see if the relevant image file resides on a local computing device in a cache. If not, the emoticon retriever 406 parses an object name or a character sequence to determine an IP address of the originating computing device.

If the custom emoticon cannot be found in a local cache, then a point-to-point (P2P) retriever 412 may try to establish a P2P connection with the message sender. A P2P connection allows quick transfer of relatively large amounts of data, so this type of connection is preferred by an emoticon retriever 406 when a

custom emoticon does not reside locally. The P2P retriever 412 tries to establish a direct connection by transmission control protocol (TCP) or user datagram protocol (UDP) between the receiver's computing device and the sender's computing device. If the connection is successful, then a port is opened and the custom emoticon is retrieved across the wire. A custom emoticon file may then be saved locally and then loaded repeatedly.

However, if a P2P connection cannot be established because of a firewall or a network address translation (NAT) table, then a server retriever 414 may establish a default connection through a server connection, since both the sender and receiver are connected to at least a server that administers the real-time communication session, e.g., a server in the Internet "cloud" that allows back and forth messaging. This latter type of connection, however, may be slower than a P2P connection. A typical server pipe is limited in the number of messages it can send per minute, and the data size of the messages. Thus the server retriever 414 may transfer custom emoticon data in smaller pieces, because of limitations placed on the pipe by the server. Custom emoticons received in pieces from the sender's custom emoticons object store 310 via a server are assembled at the receiver's end and can be loaded as a file by the message receiver 318.

Once the message receiver 318 is in possession of a custom emoticon associated with a character sequence, a substitution module 408 replaces the character sequences (e.g., textual) in the message with their corresponding custom emoticons.

It should be noted that the illustrated message receiver 318 of Fig. 4 can be implemented in software, firmware, hardware or a combination of software, firmware, hardware, etc. The illustrated exemplary message receiver 318 is only

one example of software, firmware, and/or hardware that can perform the subject matter.

Exemplary User Interface Elements

Fig. 5 shows a list of custom emoticons 500. In some implementations the list of custom emoticons 500 can be part of a dialogue box for selecting one or more of the custom emoticons for editing or for insertion into a text message—in which case a selected custom emoticon from the list 500 or a corresponding assigned character sequence that maps to the custom emoticon is inserted in an appropriate location in the text message.

In one implementation, elements of a list of custom emoticons 500 can be shown in a tooltip that appears on a display when the user hovers a pointing device over a user interface element. For example, a tooltip can appear to remind the user of available custom emoticons and/or corresponding character sequences. In the same or another implementation, a tootip appears when the user points to a particular custom emoticon in order to remind of the character sequence and nickname assigned to that custom emoticon. In the same or yet another implementation, a list of custom emoticons 500 appears as a pop-down or unfolding menu that includes a dynamic list of a limited number of the custom emoticons created in a system and/or their corresponding character sequences. For example, the menu may dynamically show ten custom emoticons and when a new custom emoticon is added to the system, a pre-existing custom emoticon drops from the list (but is not necessarily deleted from the custom emoticons object store 310).

With respect to one feature of the subject matter, many types and sizes of images can be converted for use as a custom emoticon. An image selector 308 of an exemplary CEE 206 may load an image file, such as a JPEG file 502, a TIFF file 504, a GIF file 506, a BMP file 508, a PNG file 510, etc., and convert the image to a custom emoticon (e.g., 512, 514, 516, 518, 520). In one implementation, the conversion is accomplished by representing each image as a pixel array of predetermined size: "x" pixels 522 by "y" pixels 524, e.g., 19 x 19 pixels. The image may be padded (e.g., 526 and 528) out to the size of the predetermined pixel array to maintain an image aspect ratio. Each new custom emoticon can be saved in a custom emoticons object store 310 together with its associated information, i.e., a mappable character sequence and a nickname.

Fig. 6 shows a "define custom emoticons" dialogue 600 generated, for example, by a "define custom emoticons" module 304 of a user interface 302. The dialogue 600 may include an image filename entry field 602, a character sequence assignment entry field 604 and a nickname entry field 606. The information collected in the dialogue 600 is associated with a custom emoticon, e.g., from an image selector 308 and/or a pixel array generator 314, and sent to a custom emoticons object store 310.

When a user writes a real-time text message, a custom emoticon's character sequence, such as @FDAN, can be inserted along with the other text of the message. A device and/or an application, such as an exemplary CEE 206, in response to receiving a text message including the character sequence @FDAN can map to the custom emoticon associated with @FDAN and insert the custom emoticon in place of @FDAN.

Exemplary Methods

Fig. 7 shows an exemplary method 700 of sending a custom emoticon in a real-time communication without adding graphics overhead to a text part of the communication. In the flow diagram, operations are summarized in individual blocks. The operations may be performed in hardware and/or as machine-readable instructions (software or firmware) that can be executed by a processor or engine, such as an exemplary CEE 206.

At block 702, an image is converted into a pixel array suitable for use as an emoticon, for example by an image selector 308 that includes a pixel array generator 314. Uniformity of size can be imparted to images selected to be custom emoticons by representing each custom emoticon by the same size pixel grid.

At block 704, a character sequence is assigned to the pixel array, for example, by a character sequence assignor 316 of an exemplary CEE 206 soliciting the characters of the sequence from a user. The character sequence can typically be entered in the same manner as the text of a real-time communication that contains the character sequence and not only marks the place in the communication where an associated custom emoticon should be inserted but also becomes a pointer for retrieving the custom emoticon from a sender or local cache.

At block 706, the character sequence is sent within a real-time communication. A text message being used for the real-time communication may have one or more of the character sequences entered with the rest of the text of the message. In one implementation, each character sequence is also turned into an object name that can be parsed into a location identity of an associated custom

emoticon and/or into a hash that can be used as an address for finding the custom emoticon locally.

At block 708, the pixel array is sent by an exemplary CEE 206 in a second communication that is separate from the first communication to replace the character sequence in the real-time first communication with a custom emoticon.

Fig. 8 shows an exemplary method 800 of receiving a real-time communication and retrieving a custom emoticon to replace a character sequence embedded in the real-time communication. In the flow diagram, operations are summarized in individual blocks. The operations may be performed in hardware and/or as machine-readable instructions (software or firmware) that can be executed by a processor or engine, such as an exemplary CEE 206.

At block 802, online real-time communication is received, e.g., by a message receiver 318 of an exemplary CEE 206, that includes a character sequence mappable to a pixel array, that is, a custom emoticon, residing outside of or external to the communication. In other words, the real-time communication is devoid of custom emoticons as emoticon graphics occupy too much data space for the real-time communication.

At block 804, the pixel array (i.e., the custom emoticon) is retrieved using the character sequence, for example, using an emoticon retriever 406 of an exemplary CEE 206. Directly or indirectly, the character sequence points to a local or remote location of the custom emoticon. In one implementation, the character sequence is parsed on the sender's side into an object name, which is then stored in a header of a real-time message. The object name has a first aspect that can be used to determine whether the custom emoticon being retrieved already exists in a local cache on the receiver's side. The object name also has a second

aspect that can be used to retrieve the custom emoticon from an IP address of the sender, if necessary.

In one implementation an application or device practicing the exemplary method 800 attempts to find the custom emoticon on a local cache on the receiver's side, then if the custom emoticon is not available locally tries to establish a P2P connection with the sender, and if this fails, then tries to obtain the custom emoticon through the control of an intervening server.

At block 806, the character sequence mappable to the custom emoticon is replaced in the received real-time communication with the retrieved custom emoticon, for example, by a substitution module 408 of an exemplary CEE 206. In one implementation, the position for inserting and/or substituting the custom emoticon in a real-time message may have been stored in a message header by the sender. In one implementation, a blank placeholder graphic is displayed if the retrieval of a custom emoticon is delayed, while in another implementation the character sequence associated with a custom emoticon is displayed if the retrieval of a custom emoticon is delayed.

Exemplary Computing Device

Fig. 9 shows an exemplary computing device 900 suitable as an environment for practicing aspects of the subject matter, for example as a client 202, 204 for online real-time communication. The components of computing device 900 may include, but are not limited to, a processing unit 920, a system memory 930, and a system bus 921 that couples various system components including the system memory 930 to the processing unit 920. The system bus 921 may be any of several types of bus structures including a memory bus or memory

controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISAA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as the Mezzanine bus.

Exemplary computing device 900 typically includes a variety of computing device-readable media. Computing device-readable media can be any available media that can be accessed by computing device 900 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computing device-readable media may comprise computing device storage media and communication media. Computing device storage media include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computing device-readable instructions, data structures, program modules, or other data. Computing device storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device 900. Communication media typically embodies computing device-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or

changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computing device readable media.

The system memory 930 includes computing device storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 931 and random access memory (RAM) 932. A basic input/output system 933 (BIOS), containing the basic routines that help to transfer information between elements within computing device 900, such as during start-up, is typically stored in ROM 931. RAM 932 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 920. By way of example, and not limitation, Fig. 9 illustrates operating system 934, application programs 935, other program modules 936, and program data 937. Although the exemplary CEE 206 is depicted as software in random access memory 932, other implementations of an exemplary CEE 206 can be hardware or combinations of software and hardware.

The exemplary computing device 900 may also include other removable/non-removable, volatile/nonvolatile computing device storage media. By way of example only, Fig. 9 illustrates a hard disk drive 941 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 951 that reads from or writes to a removable, nonvolatile magnetic disk 952, and an optical disk drive 955 that reads from or writes to a removable, nonvolatile optical disk 956 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computing device storage media that can be used

in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 941 is typically connected to the system bus 921 through a non-removable memory interface such as interface 940, and magnetic disk drive 951 and optical disk drive 955 are typically connected to the system bus 921 by a removable memory interface such as interface 950.

The drives and their associated computing device storage media discussed above and illustrated in Fig. 9 provide storage of computing device-readable instructions, data structures, program modules, and other data for computing device 900. In Fig. 9, for example, hard disk drive 941 is illustrated as storing operating system 944, application programs 945, other program modules 946, and program data 947. Note that these components can either be the same as or different from operating system 934, application programs 935, other program modules 936, and program data 937. Operating system 944, application programs 945, other program modules 946, and program data 947 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the exemplary computing device 900 through input devices such as a keyboard 948 and pointing device 961, commonly referred to as a mouse, trackball, or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 920 through a user input interface 960 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 962 or other type of display device is

also connected to the system bus 921 via an interface, such as a video interface 990. In addition to the monitor 962, computing devices may also include other peripheral output devices such as speakers 997 and printer 996, which may be connected through an output peripheral interface 995.

The exemplary computing device 900 may operate in a networked environment using logical connections to one or more remote computing devices, such as a remote computing device 980. The remote computing device 980 may be a personal computing device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computing device 900, although only a memory storage device 981 has been illustrated in Fig. 9. The logical connections depicted in Fig. 9 include a local area network (LAN) 971 and a wide area network (WAN) 973, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computing device networks, intranets, and the Internet.

When used in a LAN networking environment, the exemplary computing device 900 is connected to the LAN 971 through a network interface or adapter 970. When used in a WAN networking environment, the exemplary computing device 900 typically includes a modem 972 or other means for establishing communications over the WAN 973, such as the Internet. The modem 972, which may be internal or external, may be connected to the system bus 921 via the user input interface 960, or other appropriate mechanism. In a networked environment, program modules depicted relative to the exemplary computing device 900, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 9 illustrates remote application programs 985 as

residing on memory device 981. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computing devices may be used.

CONCLUSION

The subject matter described above can be implemented in hardware, in software, or in firmware, or in any combination of hardware, software, and firmware. In certain implementations, the subject matter may be described in the general context of computer-executable instructions, such as program modules, being executed by a computing device or communications device. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The subject matter can also be practiced in distributed communications environments where tasks are performed over wireless communication by remote processing devices that are linked through a communications network. In a wireless network, program modules may be located in both local and remote communications device storage media including memory storage devices.

The foregoing discussion describes exemplary custom emoticons, methods of creating and transferring emoticons, and an exemplary emoticon engine. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.